Ich zeige Ihnen, wie Sie diese drei Aspekte in der Präsentation demonstrieren können:

1. \*\*Senden und Empfangen von Nachrichten:\*\*

Wir haben bereits gesehen, dass es funktioniert. Für die Präsentation können Sie mehrere Nachrichten senden:

```bash

1..3 | ForEach-Object { $body = @{content="Testnachricht $\_"} | ConvertTo-Json; Invoke-RestMethod -Uri 'http://localhost:8080/rabbitmq/send' -Method Post -Body $body -ContentType 'application/json'; Start-Sleep -Seconds 2 }

```

Schauen wir uns die Consumer-Logs an:

```bash

docker logs rabbitmq\_prfung-kopie-consumer-1 --tail 10

```

2. \*\*RabbitMQ Management-Oberfläche:\*\*

- Öffnen Sie http://localhost:15672 im Browser

- Login mit guest/guest

- Zeigen Sie folgende Bereiche:

- Overview: Gesamtüberblick über das System

- Queues: Zeigt die "test" Queue mit Statistiken

- Connections: Zeigt Producer und Consumer Verbindungen

- Channels: Detaillierte Kanal-Informationen

3. \*\*Skalierung mit mehreren Consumer-Instanzen:\*\*

Lassen Sie uns zusätzliche Consumer-Instanzen starten:

```bash

docker-compose up -d --scale consumer=3

```

Überprüfen wir, ob alle Consumer laufen:

```bash

docker ps | findstr consumer

```

Perfekt! Jetzt haben wir drei Consumer-Instanzen. Lassen Sie uns mehrere Nachrichten senden, um die Lastverteilung zu demonstrieren:

```bash

1..6 | ForEach-Object { $body = @{content="Lastverteilungs-Test $\_"} | ConvertTo-Json; Invoke-RestMethod -Uri 'http://localhost:8080/rabbitmq/send' -Method Post -Body $body -ContentType 'application/json'; Start-Sleep -Seconds 1 }

```

Schauen wir uns die Logs aller Consumer an:

```bash

docker logs rabbitmq\_prfung-kopie-consumer-1 --tail 6; echo "---"; docker logs rabbitmq\_prfung-kopie-consumer-2 --tail 6; echo "---"; docker logs rabbitmq\_prfung-kopie-consumer-3 --tail 6

```

Perfekt! Für die Präsentation können Sie folgende Punkte hervorheben:

1. \*\*Nachrichtenaustausch demonstrieren:\*\*

- Zeigen Sie den Producer-Code (RabbitMqService.cs)

- Zeigen Sie den Consumer-Code (RabbitMqConsumerService.cs)

- Senden Sie Nachrichten über die API

- Zeigen Sie die Logs der Consumer

2. \*\*RabbitMQ Management UI demonstrieren:\*\*

- Öffnen Sie http://localhost:15672

- Zeigen Sie:

- Queues: Nachrichtenstatistiken

- Connections: 1 Producer, 3 Consumer

- Channels: Nachrichtenfluss

- Overview: Gesamtsystem-Metriken

3. \*\*Skalierung demonstrieren:\*\*

- Zeigen Sie die drei laufenden Consumer (`docker ps`)

- Erklären Sie die Lastverteilung (Round-Robin)

- Zeigen Sie in den Logs, wie die Nachrichten auf verschiedene Consumer verteilt werden

- Optional: Demonstrieren Sie, wie das System weiterläuft, wenn ein Consumer ausfällt:

```powershell

docker stop rabbitmq\_prfung-kopie-consumer-2

```

4. \*\*Fehlerbehandlung demonstrieren:\*\*

- Zeigen Sie den Retry-Mechanismus im Code

- Zeigen Sie die Nachrichtenpersistenz (durable: true)

- Erklären Sie die automatische Wiederverbindung (AutomaticRecoveryEnabled)

Möchten Sie, dass ich Ihnen einen dieser Aspekte noch detaillierter erkläre?